

Asynchrony and Collusion in the N-party BAR Transfer Problem

Xavier Vilça, Oksana Denysyuk, and Luís Rodrigues

INESC-ID, Instituto Superior Técnico, Universidade Técnica de Lisboa

Abstract. The problem of reliably transferring data from a set of N_P producers to a set of N_C consumers in the BAR model, named N-party BAR Transfer (NBART), is an important building block for volunteer computing systems. An algorithm to solve this problem in synchronous systems, which provides a Nash equilibrium, has been presented in previous work. In this paper, we propose an NBART algorithm for asynchronous systems. Furthermore, we also address the possibility of collusion among the Rational processes. Our game theoretic analysis shows that the proposed algorithm tolerates certain degree of arbitrary collusion, while still fulfilling the NBART properties.

1 Introduction

Peer-to-peer networks can be used for executing computationally intensive projects, as shown by the Boinc infrastructure [1]. Building systems on this kind of networks may be quite challenging due to the existence of Byzantine processes, whose behaviour is arbitrary, and of Rational processes, which may deviate from the specified protocols if they can increase their utility. A system model that captures this variety of behaviours has been coined the BAR model [2], named after the three classes of processes (Byzantine, Altruistic, and Rational) that it explicitly considers.

Our work focuses on the particular problem of reliably transferring data from a set of N_P producers to a set of N_C consumers in the BAR model, named N-party BAR Transfer (NBART). This problem is an important building block for volunteer computing systems, since it allows volunteers to transfer intermediate or final results of the computations to another set of volunteers, after storing the data for some time. For instance, if computations are to be performed using a model such as MapReduce, mappers may invoke the NBART primitive to transfer the intermediate results to reducers.

Although an algorithm that solves this problem has already been devised for synchronous systems [3], in a peer-to-peer network it is often unrealistic to assume that there is a known upper bound for the execution time and the communication delay. With this in mind, this paper addresses the NBART problem in an asynchronous system.

Furthermore, this paper also addresses the problem of collusion, which is a real issue in peer-to-peer networks due to attacks, such as sybil and white washing. In addition to arbitrary collusion of Byzantine players, we consider that Rational processes may create collusion groups, including producers and consumers.

Related Work Since models based on traditional Game Theory assume that all processes follow the selfish strategy that maximises their utility function, they fail to account for arbitrary behaviour that may arise from Byzantine faults. In face of this limitation, traditional utility functions must be augmented to accommodate Byzantine-awareness. Additionally, alternative rules for predicting how the game will be played have also been proposed to address Byzantine behaviour.

To the best of our knowledge, the work of Eliaz et. al. [4] was the first to address the issues above, introducing the notion of k -Fault Tolerant Nash Equilibrium (k -FTNE). In this context, a profile of strategies is k -FTNE if the strategy of each player is a best response to the strategy of other players, independently of the identity of Byzantine players and the arbitrary strategy they

follow. This concept was later applied to virus inoculation games [5]. In [6], the authors discuss the limitations imposed by *regret freedom* on communication games, by proving that there are no non-trivial equilibria that provide regret-freedom strategies. Then, they propose a different approach named *regret-braving* where players are willing to obey the specified solutions basing on their expectations about the environment, and these strategies are regret-free as long as those expectations hold. In our work, we consider that players are risk-averse, that is, they always hold the expectation that Byzantine players will follow the worst possible strategy to their utility.

In practice, rational players can seek maximising their utility function by colluding with other players, i.e., forming coalitions. Therefore, the solution concepts are more robust if they account for such rational behaviour. Aumann [7] addressed this issue by defining an equilibrium as a profile of strategies where no deviating collusion strategy provides a greater utility for all players of the group. Then, Bernheim et. al. [8] introduced the notion of *coalition-proof Nash equilibrium*, where no deviations by a coalition can perform better, although they do not allow further deviations to the collusion strategy. This work was later extended to take into consideration correlated strategies [9].

The work of [10] considered the existence of processes with unexpected utilities and collusion. The authors proposed the solution concept of (k, t) -robustness, where no process can increase its utility by deviating in collusion with up to $k - 1$ other processes, regardless of the Byzantine behaviour of up to t processes. This notion is stronger than the previous models for collusion, since it accounts for arbitrary collusion where it should be true that no player performs better by deviating from the equilibrium strategy, even if that implies decreasing the utility of other players within the coalition. Unfortunately, in certain scenarios such as communication games (where players incur communication costs), it was shown that no game can be (k, t) -robust for $k, t > 0$ [11].

Additional literature relevant to our results include works on agreement in the BAR model [2,11] and data dissemination [12,13,14], which studied protocols tolerant to the BAR model and showed in which conditions those solutions provide Nash equilibriums. In [15], the authors studied the impact of altruism on a repeated game modelled by the BAR model. All these works assume repeated interactions of processes in a cooperative service. On the other hand, our paper considers one-shot games, and therefore addresses the need to provide equilibrium strategies for Rational processes to follow the specified algorithm based on incentives provided in a single instance of NBART.

Contributions The first contribution of this paper consists in an algorithm that solves NBART in asynchronous systems. We show that the proposed algorithm is correct, assuming that all non-Byzantine processes follow it, for $N_P \geq 2F_P + 1$ and $N_C \geq F_C + 1$, where F_P and F_C are upper bounds on the number of Byzantine producers and Byzantine consumers respectively. We also show that the presented algorithm obtains asymptotically optimal bit complexity in certain scenarios.

The second contribution consists in the game theoretic analysis of the proposed algorithm. Since processes incur communication costs, our algorithm cannot be (k, t) -robust [11], hence we rely on a weaker notion of Byzantine aware utility function to account for Byzantine behaviour, based on the notion proposed in [11].

Given that we cannot ensure that the players within a coalition follow the algorithm, we propose a new solution concept, which is an adaptation of k -resilience to account for collusion in the following way. We define an equilibrium as a profile of strategies σ where members of a coalition are interested in deviating from σ only if their behaviour, as observed by other processes, is equivalent to σ .

We assume that the size of each group of Rational colluding processes is bounded by a constant $N_{\mathcal{T}} = N_{\mathcal{T}}^P + N_{\mathcal{T}}^C$, where $N_{\mathcal{T}}^P$ is the number of members of the colluding group that are producers and $N_{\mathcal{T}}^C$ is the number of consumers on the same group. We show that, if $N_P \geq \max(F_P, N_{\mathcal{T}}^P) + F_P + 1$ and $N_C \geq F_C + N_{\mathcal{T}}^C + 1$, then the algorithm provides such equilibrium, implying that processes from any coalition follow a strategy that ensures that the NBART properties are fulfilled. An important consequence of this is that, in the absence of collusion, the algorithm provides a Nash equilibrium.

Paper Organisation The remainder of the paper is structured as follows. The system model and the NBART problem are defined in Section 2. The algorithm that solves NBART in the given model is presented in Section 3, along with the proofs of correctness and a simple complexity analysis. In Section 4, we perform the game theoretic analysis of the algorithm.

2 System Model

We assume an asynchronous system composed of N *processes* or *players* (we will use the term *player* only when performing the Game Theoretic analysis; in any other case, we will use the name process). Processes are connected by a fully-connected network and can communicate using reliable authenticated point-to-point communication channels [16].

We make the distinction between *identity*, *process/player*, and *coalition*. An identity is a tuple (i, pk_i, sk_i) , where i is an identifier and pk_i and sk_i are the corresponding public and private keys. There is a set of identities $\mathcal{I} = \mathcal{P} \cup \mathcal{C}$, where \mathcal{P} and \mathcal{C} are the sets of producer and consumer identities, respectively, such that $\#\mathcal{P} = N_{\mathcal{P}}$ and $\#\mathcal{C} = N_{\mathcal{C}}$. Players are the decision-making entities of our Game Theoretic analysis and are represented by a single identity. Therefore, when referring to the process that holds the identity (i, pk_i, sk_i) , we will simply refer to it as i . If $i \in \mathcal{P}$, the corresponding process is referred to as a producer, otherwise, it is called a consumer. Finally, $N_{\mathcal{P}} + N_{\mathcal{C}} = N$.

As defined by the BAR model, a player can be Altruistic (if it follows the algorithm), Byzantine (if its behaviour is arbitrary), or Rational (if it follows the strategy that maximises its utility given the expectations regarding the strategies followed by other players). We assume that Rational processes adhere to the *promptness principle* [2], in the sense that if the expected utilities of following the algorithm and deviating by delaying messages are equivalent, then processes do not deviate. It is said that a player i signs information with sk_i by invoking $s_i(data)$.

2.1 NBART Problem

The NBART Problem can be defined as follows. Each producer p produces an arbitrarily large value v_p by invoking the deterministic function $produce(p, v_p)$, such that any two non-Byzantine producers produce the same value, named the *correct value*. Consumers must consume only one value v , sent by some producer, by invoking $consume(c, v)$. The invocation of this primitive proves that, indeed, c consumes the value. To deal with Rational behaviour, we rely on the participation of an abstract entity named Trusted Observer (TO), whose function is to gather cryptographic information from the participants of each transfer and reward processes according to their observable behaviour. To assess the behaviour of each process, TO uses two predicates $hasProd(evidence, p)$ and $hasAck(evidence, c)$ that take as input the evidence produced by TO to indicate, respectively, if producer p participated in NBART and if consumer c notified the reception of the correct value. TO is said to eventually *produce evidence* about the transfer if, when $hasProd$ and $hasAck$ become true for all corresponding non-Byzantine producers and consumers, TO eventually calls the primitive $certify(TO, evidence)$ after that. With these definitions, the NBART problem is characterised by the following properties:

- **NBART 1** (*Validity*): If a non-Byzantine consumer consumes v , then v was produced by some non-Byzantine producer.
- **NBART 2** (*Integrity*): No non-Byzantine consumer consumes more than once.
- **NBART 3** (*Agreement*): No two non-Byzantine consumers consume different values.
- **NBART 4** (*Eventual Consumption*): Eventually, every non-Byzantine consumer consumes a value.
- **NBART 5** (*Evidence*): TO eventually produces evidence about the transfer.

- **NBART 6** (*Producer Certification*): If producer p is non-Byzantine, then $hasProd(evidence, p)$ eventually becomes *true*.
- **NBART 7** (*Consumer Certification*): If consumer c is non-Byzantine, then $hasAck(evidence, c)$ eventually becomes *true*.

3 Asynchronous NBART

We now describe an algorithm that solves the NBART problem in an asynchronous environment. We first provide an overview, then proceed to the detailed description of the algorithm, and we conclude with a theoretical analysis, where we prove the correctness of this solution and perform a complexity analysis in terms of message and bit complexity.

3.1 Overview of the Algorithm

The algorithm can be briefly described as follows. Each producer p owns a block (b_p) that belongs to the set of $N_{\mathcal{P}}$ blocks obtained from the value v by using Reed-Solomon codes, such that v can be retrieved from any subset of B blocks ($N_{\mathcal{P}} \geq B + F_{\mathcal{P}}$). Then, p strives to transfer b_p along with the signature of the vector that contains the hashes of all blocks to a subset of consumers denoted by $conset_p$. Each consumer c only needs to receive B correct blocks and $F_{\mathcal{P}} + 1$ signatures of the same vector of hashes to consume the value. However, c must continue to process any received information and send it to TO, which must (re-)invoke $certify(evidence)$ whenever it receives new information, in order to fulfil the property NBART-5.

3.2 Algorithm in Depth

The algorithm is depicted for producers in Alg. 1, for consumers in Alg. 2 and Alg. 3, and for TO in Alg. 4. Producers use Reed-Solomon codes to reduce the communication costs of transferring an arbitrarily large value. The value v , whose length in bits is denoted by l_v , is split into $N_{\mathcal{P}}$ blocks of size $\frac{l_v}{B}$, such that any subset of B blocks is sufficient to retrieve the original value, where $1 \leq B \leq N_{\mathcal{P}} - F_{\mathcal{P}}$ and $B < l_v$. There is a function $RS-ENC(v, N_{\mathcal{P}}, B, \omega)$ that, given the correct value v , the number of producers $N_{\mathcal{P}}$, the number of blocks B , and the word size ω , returns a vector \mathbf{v} containing the $N_{\mathcal{P}}$ blocks, where $2^{\omega} > N_{\mathcal{P}}$. Let \mathbf{h}_v denote the vector containing the hashes of each of the blocks from \mathbf{v} . The inverse function $RS-DEC(\mathbf{v}', N_{\mathcal{P}}, B, \omega, \mathbf{h}_v)$ is defined as follows: if there are at least B blocks from \mathbf{v}' whose hash is in \mathbf{h}_v , then it returns the value v ; otherwise, it returns \perp . We consider that all arithmetic operations are performed over elements of the Galois Field $GF(2^{\omega})$.

We consider that each process is unequivocally identified by an index, between 0 and $N_{\mathcal{P}} - 1$ for producers, and between 0 and $N_{\mathcal{C}} - 1$ for consumers. Each consumer c_j uses a deterministic function $prodset_{c_j}$ to determine the set of producers that are supposed to send it their blocks, defined in such a way that each consumer is related to exactly $B + F_{\mathcal{P}}$ producers (in this way distributing load among producers). A possible mapping function is the following: $prodset_{c_j} = \{p_i \in \mathcal{P} | i \in [k \dots (k + B + F_{\mathcal{P}} - 1) \bmod N_{\mathcal{P}}], k = j(B + F_{\mathcal{P}}) \bmod N_{\mathcal{P}}\}$. It is useful to define the function that establishes the inverse relation $conset_{p_i} = \{c_j \in \mathcal{C} | p_i \in prodset_{c_j}\}$ for each producer p_i . These definitions ensure that each consumer is able to receive at least B blocks from non-Byzantine producers, therefore being able to retrieve the correct value. In addition, the load is distributed across the producers such that $\forall p \in \mathcal{P} : \#conset_p = n \Rightarrow \forall p' \in \mathcal{P} \setminus \{p\} : n - 1 \leq \#conset_{p'} \leq n + 1$.

Each producer p starts by storing the set of blocks from \mathbf{v} by invoking $RS-ENC$. Note that each producer will only be required to transmit one of these blocks (each producer transmits a different block). However, each producer is still required to send \mathbf{h}_v . Therefore, each producer then sets the vector $hashes$ to \mathbf{h}_v (Alg. 1, lines 4-7). Then, p transfers its block along with \mathbf{h}_v to all consumers

of $conset_p$ in a BLOCK message (lines 8-10), while sending SUMMARY messages to the remaining consumers only containing \mathbf{h}_v (lines 11-13). Both these messages are signed with the public key of the producer. Notice that, in the BLOCK message, it is not necessary to sign the block, for the signature of the hashes already authenticates the block.

Algorithm 1: NBART ($p \in \mathcal{P}$)

```

01 upon init() do
02   blocks :=  $[\perp]^{N_{\mathcal{P}}}$ ;
03   hashes :=  $[\perp]^{N_{\mathcal{P}}}$ ;

04 upon produce( $p, v$ ) do
05   blocks := RS-ENC(value,  $N_{\mathcal{P}}, B, \omega$ );
06   forall  $i \in \mathcal{P}$  do
07     hashes[ $i$ ] := hash(blocks[ $i$ ]);
08   signature :=  $s_p(\text{BLOCK} || \text{hashes})$ ;
09   forall  $c \in conset_p$  do
10     send( $p, c, [\text{BLOCK}, \text{blocks}[p], \text{hashes}, \text{signature}]$ );
11   signature :=  $s_p(\text{SUMMARY} || \text{hashes})$ ;
12   forall  $c \in \mathcal{C} \setminus conset_p$  do
13     send( $p, c, [\text{SUMMARY}, \text{hashes}, \text{signature}]$ );

```

In turn, each consumer c keeps all the received data blocks in a vector *blocks* and the received vectors of hashes (along with the signatures) in *hashvecs*. In addition, there is a set *missing* that keeps the identities of the producers that have not yet sent any signed information. Finally, *correcthashvec* is the correct vector of hashes, that is, the vector that is sent by at least $F_{\mathcal{P}} + 1$ producers, and *correctproducers* stores, for each producer, the value \perp if it has not yet sent any message, or the signature of the message sent by the producer.

Each consumer uses the functions *verifysig*(i, d) and *verifyhash*(b, h) to verify the signature by i of d and the hash of b when compared to h , respectively. Consumer c is in one of three states: *init*, *gotHashes*, and *consumed*. c is in state *init* when *hashvecs* does not contain a majority ($F_{\mathcal{P}} + 1$) of identical vectors of hashes. The function *minimumHashes* (Alg. 2, lines 8-12) marks the transition between *init* and *gotHashes*, by setting *correcthashvec* to a non-null value, when the required majority of hashes is gathered by c . Procedure *consume-and-report* (lines 16-23) makes the transition from *gotHashes* to *consumed* when the consumer gathers at least B correct blocks and, therefore, the invocation of *RS-DEC* returns a non-null value. In this case, the consumer consumes the value (line 19) and prepares a report intended to TO (lines 20-23), which is sent by invoking the procedure *report* (lines 13-15). This report contains the vector *correcthashvec* and the signature of all the producers that already sent correct messages to c , i.e., messages that contained *correcthashvec*.

Whenever a consumer c receives a BLOCK message from a producer that belongs to $missing \cap prodset_c$ (Alg. 3, line 1), c removes p from *missing* if the signature is valid (lines 2-3) and, according to its state, performs one of the following actions: i) If c is still in state *init*, then it stores the received information in the appropriate vectors and invokes *minimumHashes* (lines 4-8), in order to verify if it has already gathered a majority of identical vectors of hashes. If that is the case, then c invokes *consume-and-report* (lines 9-10). ii) If c is in state *gotHashes*, then it adds the received vector of hashes along with the signature to *hashvecs*, stores the block, and invokes *consume-and-report* (lines 11-15). iii) If c is in state *consumed*, then it adds the signature of the producer to *correctproducers* and reports the information received from producers to TO (lines 16-18).

An almost identical approach is followed by c whenever it receives a SUMMARY message, aside from the fact that in this case c does not expect to receive any block (lines 19-33).

Algorithm 2: NBART ($c \in \mathcal{C}$): Part I

```
01 upon init do
02   value :=  $\perp$ ;
03   correcthashvec :=  $\perp$ ;
04   hashvecs :=  $[\perp]^{N_{\mathcal{P}}}$ ;
05   blocks :=  $[\perp]^{N_{\mathcal{P}}}$ ;
06   missing :=  $\mathcal{P}$ ;
07   correctproducers :=  $[\perp]^{N_{\mathcal{P}}}$ ;

08 function minimumHashes(hashvecs) is
09   if  $\exists h : \#\{p | \text{hashvecs}[p] = \langle h, * \rangle\} \geq F_{\mathcal{P}} + 1$  then
10     return h;
11   else
12     return  $\perp$ ;

13 procedure report is
14   signature :=  $s_c(\text{REPORT} || \text{correcthashvec} || \text{correctproducers})$ ;
15   send(c, TO, [REPORT, correcthashvec, correctproducers, signature]);

16 procedure consume-and-report is
17   value := RS-DEC(blocks,  $N_{\mathcal{P}}$ , B,  $\omega$ , correcthashvec);
18   if value  $\neq \perp$  then
19     consume(c, value);
20     forall  $p \in \mathcal{P}$  do
21       if hashvecs[p] =  $\langle \text{correcthashvec}, \text{signature} \rangle$  then
22         correctproducers[p] := signature;
23     report ();
```

The trusted observer only waits for REPORT messages from consumers to include all the received information in the array *evidence* (lines 3-5). In addition, TO repeatedly tries to produce the evidence about the transfer whenever it receives new information (line 6).

3.3 Predicates

We now define the predicates *hasProd* and *hasAck*. It is said that producer *p* is *certified* by consumer $c \in \text{conset}_p$ iff $\text{evidence}[c] = \langle \mathbf{h}_v, \text{report} \rangle$ and $\text{report}[p] = s_p(\text{BLOCK}, \mathbf{h}_v)$. We say that producer *p* is certified by consumer $c \in \mathcal{C} \setminus \text{conset}_p$ iff $\text{evidence}[c] = \langle \mathbf{h}_v, \text{report} \rangle$ and $\text{report}[p] = s_p(\text{SUMMARY}, \mathbf{h}_v)$. Let $\bar{\mathcal{P}} \subseteq \mathcal{P}$ and $\bar{\mathcal{C}} \subseteq \mathcal{C}$ be the greatest sets that fulfil the following conditions: i) for each $p \in \bar{\mathcal{P}}$ and $c \in \bar{\mathcal{C}}$, *p* is certified by *c*; and ii) for each $c \in \bar{\mathcal{C}}$, *c* invokes *consume*(*c*, *v*).

With this in mind, we now define the predicates as follows:

- For the predicates to be true for any process, $\#\bar{\mathcal{P}} \geq N_{\mathcal{P}} - F_{\mathcal{P}}$ and $\#\bar{\mathcal{C}} \geq N_{\mathcal{C}} - F_{\mathcal{C}}$;
- *hasProd*(*evidence*, *p*) is true iff $p \in \bar{\mathcal{P}}$;
- *hasAck*(*evidence*, *c*) is true iff $c \in \bar{\mathcal{C}}$.

3.4 Correctness

In this section, the correctness of the above algorithm is proven in an asynchronous environment, assuming that $N_{\mathcal{P}} \geq 2F_{\mathcal{P}} + 1$, $N_{\mathcal{C}} \geq F_{\mathcal{C}} + 1$, and that all non-Byzantine processes follow the algorithm. In the following two lemmas, we start by showing that the consumers eventually gather enough information to consume the correct value.

Lemma 1. *For each non-Byzantine consumer $c \in \mathcal{C}$, minimumHashes eventually returns exactly one vector $\mathbf{h}^* \neq \perp$ and $\mathbf{h}^* = \mathbf{h}_v$.*

Algorithm 3: NBART ($c \in \mathcal{C}$): Part II

```
01 upon deliver( $p, c, [\text{BLOCK}, \text{pblock}, \text{phashes}, \text{msgsig}] \wedge p \in \text{missing} \cap \text{prodset}_c$ ) do
02   if verifysig( $p, \text{BLOCK} || \text{phashes}, \text{msgsig}$ ) then
03      $\text{missing} := \text{missing} \setminus \{p\}$ ;
04     if verifyhash( $\text{pblock}, \text{phashes}[p]$ ) then
05       if correcthashvec =  $\perp$  then
06          $\text{hashvecs}[p] := \langle \text{phashes}, \text{msgsig} \rangle$ ;
07          $\text{blocks}[p] := \text{pblock}$ ;
08         correcthashvec := minimumHashes(hashvecs);
09         if correcthashvec  $\neq \perp$  then
10           consume-and-report ();
11         else if value =  $\perp$  then
12           if phashes = correcthashvec then
13              $\text{hashvecs}[p] := \langle \text{phashes}, \text{msgsig} \rangle$ ;
14              $\text{blocks}[p] := \text{pblock}$ ;
15             consume-and-report ();
16           else if phashes = correcthashvec then
17              $\text{correctproducers}[p] := \text{msgsig}$ ;
18             report ();

19 upon deliver( $p, c, [\text{SUMMARY}, \text{phashes}, \text{msgsig}] \wedge p \in \text{missing} \cap \mathcal{P} \setminus \text{prodset}_c$ ) do
20   if verifysig( $p, \text{SUMMARY} || \text{phashes}, \text{msgsig}$ ) then
21      $\text{missing} := \text{missing} \setminus \{p\}$ ;
22     if correcthashvec =  $\perp$  then
23        $\text{hashvecs}[p] := \langle \text{phashes}, \text{msgsig} \rangle$ ;
24       correcthashvec := minimumHashes(hashvecs);
25       if correcthashvec  $\neq \perp$  then
26         consume-and-report ();
27     else if value =  $\perp$  then
28       if phashes = correcthashvec then
29          $\text{hashvecs}[p] := \langle \text{phashes}, \text{msgsig} \rangle$ ;
30         consume-and-report ();
31     else if phashes = correcthashvec then
32        $\text{correctproducers}[p] := \text{msgsig}$ ;
33       report ();
```

Proof. Every non-Byzantine consumer receives \mathbf{h}_v from all non-Byzantine producers, eventually. Since $N_{\mathcal{P}} \geq 2F_{\mathcal{P}} + 1$, only the vector \mathbf{h}_v can be sent by $F_{\mathcal{P}} + 1$ producers. Therefore, *minimumHashes* only returns a non-null vector \mathbf{h}^* if $\mathbf{h}^* = \mathbf{h}_v$ and this occurs eventually. Also, when *correcthashvec* becomes non-null, c never invokes *minimumHashes* again. \square

Lemma 2. *For each consumer $c \in \mathcal{C}$, c eventually invokes $\text{consume}(c, v)$, and only once.*

Proof. It follows from Lemma 1 that, for each non-Byzantine consumer c , *correcthashvec* is eventually set to \mathbf{h}_v , and c eventually starts invoking *consume-and-report*. By the fact that producers send their blocks to all consumers of *conset*, and by the conditions $N_{\mathcal{P}} \geq B + F_{\mathcal{P}}$ and $\#\text{prodset}_c = B + F_{\mathcal{P}}$, c eventually receives B blocks, correct according to \mathbf{h}_v . Hence, RS-ENC eventually returns v , and only v by the property of non-collision of hash functions. A trivial inspection of the algorithm shows that, once *value* is set to $v \neq \perp$, c consumes v and never invokes *consume-and-report* again. \square

Lemma 3. *For each non-Byzantine producer p and each non-Byzantine consumer $c \in \text{conset}_p$, eventually c certifies p .*

Proof. According to the algorithm, p always sends a BLOCK message containing its block and $s_p(\text{BLOCK} || \mathbf{h}_v)$ to all $c \in \text{conset}_p$, whereas p sends a SUMMARY message to all $c \in \mathcal{C} \setminus \text{conset}_p$, containing $s_p(\text{SUMMARY} || \mathbf{h}_v)$. If c receives this information when it is still in one of the states *init* and *gotHashes*, then, by Lemma 2, c eventually sends a report to TO containing this information. If c is already in state *consumed*, then c immediately sends the report containing this information when it receives the message from p . Either way, c eventually certifies p . \square

Algorithm 4: NBART (trusted observer TO)

```
01 upon init do  
02   evidence :=  $[\perp]^{N_C}$ ;  
  
03 upon deliver(c, TO, [REPORT, hashesvec, producers, signature]) do  
04   if verifySig(c, REPORT||hashesvec||producers, signature) then  
05     evidence[c] := (hashesvec,producers);  
06     certify(TO, evidence);
```

Lemma 4. *There exist sets $\bar{\mathcal{P}}$ and $\bar{\mathcal{C}}$ of non-Byzantine producers and non-Byzantine consumers, respectively, such that: i) $\#\bar{\mathcal{P}} \geq N_P - F_P$ and $\#\bar{\mathcal{C}} \geq N_C - F_C$; ii) for each $p \in \bar{\mathcal{P}}$ and $c \in \bar{\mathcal{C}}$, c eventually certifies p ; and iii) for each $c \in \bar{\mathcal{C}}$, c eventually invokes $\text{consume}(c,v)$, where v is the correct value.*

Proof. i) follows from the fact that there are $N_P - F_P$ non-Byzantine producers and $N_C - F_C$ non-Byzantine consumers; ii) follows from Lemma 3; and iii) follows from Lemma 2. \square

The next theorem concludes the proofs of correctness by showing that each NBART property is fulfilled by the presented algorithm.

Theorem 5. *The proposed algorithm solves NBART in an asynchronous environment, assuming that all non-Byzantine processes follow the algorithm.*

Proof. The proof is performed individually for each property:

- (*Validity*): By Lemmas 1 and 2 and by the non-collision property of hash functions, c consumes the correct value, which is produced by all non-Byzantine producers.
- (*Integrity*): Follows from Lemma 2.
- (*Agreement*): It follows directly from *Validity* and the fact that all non-Byzantine producers send a block corresponding to the same value.
- (*Eventual Consumption*): Follows from Lemma 2.
- (*Evidence*): TO invokes *certify*(*evidence*) whenever it receives new information, either from producers or consumers. Thus, whenever *hasProd*(*evidence*, p) and *hasAck*(*evidence*, c) become true for each non-Byzantine producer p and non-Byzantine consumer c respectively, TO invokes *certify*(*evidence*).
- (*Producer and Consumer Certification*): Follows from Lemma 4.

\square

3.5 Complexity Analysis

The algorithm is evaluated in terms of message and bit complexity. The message complexity is $O(N_P N_C)$ due to N_C messages sent by each producer that contain the signature of \mathbf{h}_v . However, since the value may be arbitrarily large, the size of each message may vary significantly, so it is interesting to also evaluate the number of bits exchanged, that is, the bit complexity. For this analysis, let l_v , l_s , and l_h denote the bit length of the value, a signature and an hash. The bit complexity is $O(N_C(B + F_P)\frac{l_v}{B} + N_P N_C(l_s + N_P l_h))$. Notice that, if $B \geq O(F_P)$ and $l_v \gg l_s, l_h$, then the bit complexity is $O(N_C)$, which is asymptotically optimal, since there must be at least a value transfer per consumer.

4 Game Theoretic Analysis

The purpose of this analysis is to show that it is in every Rational process interest to follow the algorithm. We take into consideration some degree of arbitrary collusion.

4.1 Definitions

The algorithm is modelled as a coalitional game $\Gamma = (\mathcal{I}, \mathcal{T}, \Sigma_{\mathcal{I}}, (\succeq_t)_{t \in \mathcal{T}}, (u_i)_{i \in \mathcal{I}})$:

- $\mathcal{I} = \mathcal{P} \cup \mathcal{C} \cup \{\text{TO}\}$ is the set of players.
- \mathcal{T} is the set of non-empty subsets of $\mathcal{I} \setminus \{\text{TO}\}$, which contains all the possible coalitions. Each coalition $t \in \mathcal{T}$ may contain simultaneously producers and consumers, represented by $t_{\mathcal{P}} = t \cap \mathcal{P}$ and $t_{\mathcal{C}} = t \cap \mathcal{C}$, respectively.
- $\Sigma_{\mathcal{I}}$ is a set containing all the profile of pure strategies $\sigma_{\mathcal{I}}$ followed by all players of \mathcal{I} . Σ_t for $t \in \mathcal{T}$ denotes the set of all collusion strategies the players of t may follow.
- \succeq_t is a preference relation on $\Sigma_{\mathcal{I}} \times \Sigma_{\mathcal{I}}$. We assume that \succeq_t is transitive and reflexive. We can define the relation of strict preference \succ_t as: for any two profiles of strategies $\sigma_{\mathcal{I}}^*, \sigma_{\mathcal{I}}' \in \Sigma_{\mathcal{I}}$, $\sigma_{\mathcal{I}}^* \succ_t \sigma_{\mathcal{I}}'$ iff $\neg(\sigma_{\mathcal{I}}' \succeq_t \sigma_{\mathcal{I}}^*)$. If $\sigma_{\mathcal{I}}^* \succ_t \sigma_{\mathcal{I}}'$, then all the players of t will always follow $\sigma_{\mathcal{I}}^*$ over $\sigma_{\mathcal{I}}'$.
- u_i is the utility function of each player $i \in \mathcal{I}$, defined as $u_i(\sigma_{\mathcal{I}}) = \beta_i(\sigma_{\mathcal{I}}) - \alpha_i(\sigma_{\mathcal{I}})$, where $\beta_i(\sigma_{\mathcal{I}})$ are the benefits and $\alpha_i(\sigma_{\mathcal{I}})$ the costs i incurs when players obey $\sigma_{\mathcal{I}}$.

Sometimes, we will denote the composition of two profiles σ_A and σ_B as $\sigma_{A \cup B} = (\sigma_A, \sigma_B)$, where A and B are any two disjoint sets of players. Conversely, $u_i(\sigma_A, \sigma_B)$ is equivalent to $u_i(\sigma_{A \cup B})$. Each producer p obtains a benefit β_p iff *hasProd*(evidence, p) eventually becomes true, whereas each consumer c obtains a benefit β_c iff *hasAck*(evidence, c) eventually becomes true. It is assumed that for all $p \in \mathcal{P}$, $\beta_p > \alpha_p(\sigma_{\mathcal{I}})$, and for all $c \in \mathcal{C}$, $\beta_c > \alpha_c(\sigma_{\mathcal{I}})$, where $\sigma_{\mathcal{I}}$ is the profile of strategies where all players follow the algorithm.

A coalition t is said to be Rational if the preference relation \succeq_t fulfils the following condition:

$$\forall i \in t \forall \sigma_{\mathcal{I}} \in \Sigma_{\mathcal{I}}, \sigma_t^* \in \Sigma_t u_i(\sigma_{\mathcal{I}}) \geq u_i(\sigma_t^*, \sigma_{\mathcal{I} \setminus t}) \Rightarrow (\sigma_t, \sigma_{\mathcal{I} \setminus t}) \succeq_t (\sigma_t^*, \sigma_{\mathcal{I} \setminus t}).$$

We assume that the same relation holds, by only replacing \geq for $>$ and \succeq_t for \succ_t . It follows that if $\#t = 1$ and the only player $i \in t$ is Rational, then for any two profiles of strategies $\sigma_{\mathcal{I}}^*, \sigma_{\mathcal{I}}' \in \Sigma_{\mathcal{I}}$, $\sigma_{\mathcal{I}}^* \succeq_t \sigma_{\mathcal{I}}'$ iff $u_i(\sigma_{\mathcal{I}}^*) \geq u_i(\sigma_{\mathcal{I}}')$. On the contrary, if $\#t = 1$ and the player $i \in t$ is Altruistic, then t is also said to be Altruistic and it is true that $(\sigma_t, \sigma_{\mathcal{I} \setminus t}^*) \succ_t (\sigma_{\mathcal{I}}^*)$ for all $\sigma_{\mathcal{I}}^* \in \Sigma_{\mathcal{I}}$ and considering that $\sigma_{\mathcal{I}}$ denotes the profile of strategies where all players follow the algorithm. In any other case, t is Byzantine, implying that \succeq_t is arbitrary due to the Byzantine behaviour of some player from t . It is important to notice that, if t is Byzantine, then all players of t are also considered to be Byzantine, even if some of them have Rational intentions. A coalition t is said to be a producer ($t \in \mathcal{T}_{\mathcal{P}}$) if $t_{\mathcal{P}} \neq \emptyset$ and it is said to be a consumer ($t \in \mathcal{T}_{\mathcal{C}}$) if $t_{\mathcal{C}} \neq \emptyset$. The purpose of these definitions is to model scenarios of arbitrary collusion where, for instance, a producer p never executes any local function to produce the value. Instead, it requests the hash of the blocks to other player i , signs this information, and sends it to i . Then, i may transfer the block and the signature of p to all the consumers that expect this information, as if it were sent by p .

For simplicity, we model Byzantine behaviour as a single coalition composed by up to $F_{\mathcal{P}} + F_{\mathcal{C}}$ players. We consider an arbitrary number of non-Byzantine coalitions, as long as each coalition is never composed by more than $N_{\mathcal{T}}^{\mathcal{P}}$ producers and $N_{\mathcal{T}}^{\mathcal{C}}$ consumers. The distinction between producers and consumers will allow us a more refined analysis of the bounds on the minimum number of producers and consumers. If we only considered a single parameter, the bounds would be stricter than necessary. As it will be shown later, we now require the following conditions to hold for the algorithm to be tolerant to collusion: $N_{\mathcal{P}} \geq \max(F_{\mathcal{P}}, N_{\mathcal{T}}^{\mathcal{P}}) + F_{\mathcal{P}} + 1$ and $N_{\mathcal{C}} \geq F_{\mathcal{C}} + N_{\mathcal{T}}^{\mathcal{C}} + 1$.

4.2 Expected Utility and Solution Concept

We use the notion of Byzantine-aware utility function for risk-averse players introduced in [11]. An improvement of this work for models where players may be risk-seekers is left for future work. Let \mathcal{F}_P and \mathcal{F}_C denote the set of Byzantine producers and consumers, respectively, and let $\pi_P \in \Pi_P$ and $\pi_C \in \Pi_C$ be the corresponding profiles of strategies. Let us denote by $\sigma_{\mathcal{I} \setminus \mathcal{F}, \pi_C, \pi_C}$ the profile of strategies where all non-Byzantine players follow the strategy specified by $\sigma_{\mathcal{I}}$, Byzantine producers follow the strategies of π_P and Byzantine consumers obey the strategies of π_C . The expected utility of each player $i \in \mathcal{I} \setminus \mathcal{F}$ is defined as follows:

$$\bar{u}_i(\sigma_{\mathcal{M}}) = \min_{\mathcal{F}_P: \#\mathcal{F}_P \leq F_P, \mathcal{F}_C: \#\mathcal{F}_C \leq F_C} \circ \min_{\pi_P \in \Pi_P, \pi_C \in \Pi_C} \circ u_i(\sigma'_{M \setminus \mathcal{F}, \pi_C, \pi_C}). \quad (1)$$

Recall that, since we consider communication costs, a solution concept as strong as (k, t) -robustness is impossible in our case. To overcome this impossibility result, we use the concept of k -resilience combined with the Byzantine aware utility function defined above. However, we still cannot ensure that no player from a coalition t can increase its utility regardless of whether some other player obtains a lower utility or not. What we intend to show is that, regardless of the preferred collusion strategy of each coalition, the chosen strategies fulfil the NBART properties.

In order to formalise this intuition, we define the *observable behaviour* of each coalition $t \in \mathcal{T}$ for the profile of strategies σ_t as a multi-set of events triggered in each player $i \in \mathcal{I} \setminus t$ that are influenced by σ_t , which we denote by $\phi_i(\sigma_t)$. For any player $i \in \mathcal{I} \setminus t$, the delivery of a message sent by some player $j \in t$ is an event. In addition, there are two events triggered in TO, namely *produce*(p, v) for each $p \in t_P$ and *consume*(c, v) for each $c \in t_C$. Henceforth, the meaning of a producer producing a value or a consumer consuming a value is that the corresponding event is eventually triggered in TO.

We say that collusion profile $\sigma_t^* \in \Sigma_t$ is compliant with the profile $\sigma_{\mathcal{I}} = (\sigma_t, \sigma_{\mathcal{I} \setminus t})$ if $\forall i \in \mathcal{I} \setminus t \phi_i(\sigma_t^*) = \phi_i(\sigma_t)$. The set of profiles of strategies compliant with $\sigma_{\mathcal{I}}$ is denoted by $\Sigma_t(\sigma_{\mathcal{I}})$, where $\sigma_t \in \Sigma_t(\sigma_{\mathcal{I}})$. The solution concept we use in this work, named *n collusion tolerance* (n -cotolerance), is similar to the concept of k -resilience, aside from the fact that we do not require that players in collusion follow the algorithm exactly; only that they follow a profile of strategies from $\Sigma_t(\sigma_{\mathcal{I}})$. More precisely:

Definition 6. For any $n \in \mathbb{N}$, a profile of strategies $\sigma_{\mathcal{I}}$ is n -cotolerant iff for all $t \in \mathcal{T}$ such that $\#t \leq n$, for all $\sigma_t^* \in \Sigma_t(\sigma_{\mathcal{I}})$ such that $(\sigma_t^*, \sigma_{\mathcal{I} \setminus t}) \succeq_t \sigma_{\mathcal{I}}$, and for all $\sigma_t' \in \Sigma_t \setminus \Sigma_t(\sigma_{\mathcal{I}})$, $(\sigma_t^*, \sigma_{\mathcal{I} \setminus t}) \succ_t (\sigma_t', \sigma_{\mathcal{I} \setminus t})$.

The above definition is generic and may be of independent interest. In order to apply it to the NBART problem, we additionally need to capture the distinction between producers and consumers. Therefore, we introduce two parameters $x, y \in \mathbb{N}$, that establish the limit on the number of producers and consumers within the coalition respectively, such that $n \geq x, y$ and $n \leq x + y$. With this definition, if $n = 1$, then there is no collusion among non-Byzantine players. Henceforth, we will say that a profile of strategies $\sigma_{\mathcal{I}}$ is (n, x, y) -cotolerant iff it is n -cotolerant, $n \geq x, y$ and $n \leq x + y$, and for all $t \in \mathcal{T}$ $\#t_P \leq x$ and $\#t_C \leq y$.

4.3 Tolerance to Collusion

The purpose of this section is twofold: i) show that, considering that $\sigma_{\mathcal{I}}$ denotes the profile of strategies where all players follow the algorithm, for any combination of Byzantine and Rational collusions, and any coalition t , if all players of t follow a profile of strategies from $\Sigma_t(\sigma_{\mathcal{I}})$, then the NBART properties are fulfilled; and ii) show that any profile of strategies $\sigma_t^* \in \Sigma_t$ is preferable to σ_t only if $\sigma_t^* \in \Sigma_t(\sigma_{\mathcal{I}})$. The proofs of this section rely on the assumption that $N_P \geq \max(F_P, N_{\mathcal{T}}^P) + F_P + 1$ and $N_C \geq F_C + N_{\mathcal{T}}^C + 1$.

We find it useful to identify the following corollary that states that in any coalition t , $produce(p, v)$ must be invoked for all $p \in t_{\mathcal{P}}$, which follows from the fact that $produce(p, v) \in \phi_{\text{TO}}(\sigma_t)$.

Corollary 7. *For any $t \in \mathcal{T}_{\mathcal{P}}$, if t follows a profile of strategies from $\Sigma_t(\sigma_{\mathcal{I}})$, then for each $p \in t_{\mathcal{P}}$, p invokes $produce(p, v)$.*

Let $\epsilon : \Sigma_t \rightarrow \mathcal{E}^{N_{\mathcal{C}}}$ be a function that for each profile $\sigma_{\mathcal{I}}^*$ returns an instance of the data structure evidence $\in \mathcal{E}^{N_{\mathcal{C}}}$ stored by TO when it produces evidence about the transfer, by replacing any entrance corresponding to a Byzantine player by the value \perp , i.e., if $c \in \mathcal{F}_{\mathcal{C}}$ and $e = \epsilon(\sigma_{\mathcal{I}}^*)$, then $e[c] = \perp$, and if $p \in \mathcal{F}_{\mathcal{P}}$, then $e[c][p] = \perp$ for all $c \in \mathcal{C} \setminus \mathcal{F}_{\mathcal{C}}$.

We state the following proposition that ϵ depends only on the observable behaviour of each player:

Proposition 8. *For any $t \in \mathcal{T}$ and for any profile $\sigma_{\mathcal{I}}^* \in \Sigma_{\mathcal{I}}$, if $\phi_{\text{TO}}(\sigma_{\mathcal{I}}^*) = \phi_{\text{TO}}(\sigma_{\mathcal{I}})$, then $\epsilon(\sigma_{\mathcal{I}}^*) = \epsilon(\sigma_{\mathcal{I}})$.*

We show in the following theorem that if each coalition t follows a strategy from $\Sigma_t(\sigma_t^*)$, then the algorithm tolerates collusion. Fix any arbitrary $f \in \mathcal{F}$ such that $\#f_{\mathcal{P}} \leq F_{\mathcal{P}}$ and $\#f_{\mathcal{C}} \leq F_{\mathcal{C}}$. By assumption, $N_{\mathcal{P}} \geq \max(F_{\mathcal{P}}, N_{\mathcal{T}}^{\mathcal{P}}) + F_{\mathcal{P}} + 1$ and $N_{\mathcal{C}} \geq F_{\mathcal{C}} + N_{\mathcal{T}}^{\mathcal{C}} + 1$, and let l denote an arbitrary partition of $\mathcal{I} \setminus (\{\text{TO}\} \cup f)$ such that, for any $t \in l$, $\#t_{\mathcal{P}} \leq N_{\mathcal{T}}^{\mathcal{P}}$ and $\#t_{\mathcal{C}} \leq N_{\mathcal{T}}^{\mathcal{C}}$. We use the notation $\#(e, m)$ to denote the frequency of element e in the multi-set m .

Theorem 9. *For some arbitrary partition l , and for any $\sigma_{\mathcal{I}}^* = ((\sigma_t^*)_{t \in l}, \sigma_t^* \in \Sigma_t(\sigma_{\mathcal{I}}), (\pi_p)_{p \in f_{\mathcal{P}}}, (\pi_c)_{c \in f_{\mathcal{C}}})$, if all players follow $\sigma_{\mathcal{I}}^*$, then the NBART properties are fulfilled.*

Proof. Notice that, in this scenario, it is also true that: 1) $N_{\mathcal{P}} \geq 2F_{\mathcal{P}} + 1$ and 2) $N_{\mathcal{C}} \geq F_{\mathcal{C}} + 1$. Let us fix some arbitrary $t \in \mathcal{T}_{\mathcal{C}} \cap l$ and $c \in t_{\mathcal{C}}$. The correctness is proved for each of the NBART properties:

- (*Validity*): $consume(c, v) \in \phi_{\text{TO}}(\sigma_t^*)$, where v must be a value for which there are $F_{\mathcal{P}} + 1$ signatures of \mathbf{h}_v , otherwise $\phi_{\text{TO}}(\sigma_{\mathcal{I}}^*) \neq \phi_{\text{TO}}(\sigma_{\mathcal{I}})$ and $\sigma_t^* \notin \Sigma_t(\sigma_{\mathcal{I}})$. By 1) and Corollary 7, there is only one value that fulfils these restrictions, which is the value produced by all non-Byzantine producers.
- (*Integrity*): Since the players of t follow σ_t^* and $\sigma_t^* \in \Sigma_t(\sigma_{\mathcal{I}})$, $\#(consume(c, v), \phi_{\text{TO}}(\sigma_t^*)) = 1$.
- (*Agreement*): It follows directly from *Validity* and Corollary 7.
- (*Eventual Consumption*): Since $\phi_c(\sigma_{\mathcal{I}}^*) = \phi_c(\sigma_{\mathcal{I}})$, t receives blocks from all the non-Byzantine producers from $prodset_c \setminus t_{\mathcal{P}}$. If $\#(prodset_c \setminus t_{\mathcal{P}}) \geq F_{\mathcal{P}} + B$, then c eventually gathers B blocks corresponding to the correct value, otherwise, $\#t_{\mathcal{P}} \geq 1$ and by Corollary 7 some producer of t produces the value. In either case, by the definition of $\Sigma_t(\sigma_{\mathcal{I}})$, c must invoke $consume(c, v)$.
- (*Evidence*): It follows from the fact that TO is Altruistic.
- (*Producer and Consumer Certification*): By the definition of $\Sigma_t(\sigma_t^*)$, $\phi_{\text{TO}}(\sigma_{\mathcal{I}}^*) = \phi_{\text{TO}}(\sigma_{\mathcal{I}})$. By Theorem 5 and by 1) and 2), if all players follow $\sigma_{\mathcal{I}}^*$, then the properties NBART 6-7 are fulfilled for $e = \epsilon(\sigma_{\mathcal{I}})$. It follows from Proposition 8 that $\epsilon(\sigma_{\mathcal{I}}^*) = e$. Since the value of the predicates only depends on e , then these properties also hold in this new scenario.

□

We now provide the proofs that the profile of strategies $\sigma_{\mathcal{I}}$ where all players follow the algorithm is $(N_{\mathcal{T}}^{\mathcal{P}} + N_{\mathcal{T}}^{\mathcal{C}}, N_{\mathcal{T}}^{\mathcal{P}}, N_{\mathcal{T}}^{\mathcal{C}})$ -cotolerant for $N_{\mathcal{P}} \geq \max(F_{\mathcal{P}}, N_{\mathcal{T}}^{\mathcal{P}}) + F_{\mathcal{P}} + 1$ and $N_{\mathcal{C}} \geq F_{\mathcal{C}} + N_{\mathcal{T}}^{\mathcal{C}} + 1$. The following two lemmas show that, for each $t \in \mathcal{T}$ the expected benefit is 0 for all $i \in t$, whenever players of t follow a profile of strategies from $\Sigma_t \setminus \Sigma_t(\sigma_{\mathcal{I}})$. Recall that we assume that the players are risk averse. Therefore, the analysis is done assuming worst case Byzantine behaviour.

Lemma 10. For any $t \in \mathcal{T}_C$, let $\sigma'_t \in \Sigma_t \setminus \Sigma_t(\sigma_{\mathcal{I}})$ be any profile of strategies where t does not ensure that for all $c \in t_C$ and $p \in \mathcal{P}$, c certifies p and invokes $\text{consume}(c, v)$, and, for each, $p \in t_P$ p invokes $\text{produce}(p, v)$. Then, for all $i \in t$, $\bar{\beta}_i(\sigma'_t, \sigma_{\mathcal{I} \setminus t}) = 0$.

Proof. Assume worst case Byzantine behaviour. If $n \geq 1$ producers are not certified by all consumers of t , those n producers are certified by less than $N_C - F_C$ consumers. Since $\#t_P < N_P - F_P$, $\#\bar{P} \leq N_P - n - F_P < N_P - F_P$. Conversely, if consumers from t_C do not consume the correct value, then it is true that $\#\bar{C} < N_C - F_C$, due to the fact that $\#t_C < N_C - F_C$. By the definition of the predicates, for all $p \in t_P$ and $c \in t_C$, $\text{hasProd}(\text{evidence}, p)$ and $\text{hasAck}(\text{evidence}, c)$ are false. Therefore, for all $i \in t$ $\bar{\beta}_i(\sigma'_t, \sigma_{\mathcal{I} \setminus t}) = 0$. \square

Lemma 11. For any $t \in \mathcal{T}$, let $\sigma'_t \in \Sigma_t \setminus \Sigma_t(\sigma_{\mathcal{I}})$. Then, for all $i \in t$, $\bar{\beta}_i(\sigma'_t, \sigma_{\mathcal{I} \setminus t}) = 0$.

Proof. Assume worst case Byzantine behaviour. By the definition of $\Sigma_t(\sigma_{\mathcal{I}})$, there exists $j \in \mathcal{I} \setminus (\mathcal{F} \cup t)$ such that $\phi_j(\sigma'_t) \neq \phi_j(\sigma_t)$, which implies that not all expected events are triggered in j for some player $i \in t$, some consumer does not consume the correct value, or some producer does not produce the correct value. If j is a consumer or a producer, then it follows directly from Lemma 10 that, for all $i \in t$, $\bar{\beta}_i(\sigma'_t, \sigma_{\mathcal{I} \setminus t}) = 0$. If j is TO, then either 1) i is a producer, and i is not certified by some consumer or does not produce the value; or 2) i is a consumer, and i does not certify some player or does not consume the value. In both cases, by Lemma 10, it is true that for all $i \in t$, $\bar{\beta}_i(\sigma'_t, \sigma_{\mathcal{I} \setminus t}) = 0$. \square

The following theorem concludes that the proposed algorithm is $(N_{\mathcal{T}}^P + N_{\mathcal{T}}^C, N_{\mathcal{T}}^P, N_{\mathcal{T}}^C)$ -cotolerant.

Theorem 12. Let $\sigma_{\mathcal{I}} \in \Sigma_{\mathcal{I}}$ denote the profile of strategies where all players follow the algorithm. Then, $\sigma_{\mathcal{I}}$ is $(N_{\mathcal{T}}^P + N_{\mathcal{T}}^C, N_{\mathcal{T}}^P, N_{\mathcal{T}}^C)$ -cotolerant.

Proof. Let $t \in \mathcal{T}$ be any coalition such that $\#t_P \leq N_{\mathcal{T}}^P$ and $\#t_C \leq N_{\mathcal{T}}^C$. By Theorem 9, for all $p \in t_P$, $\bar{\beta}_p(\sigma_{\mathcal{I}}) = \beta_p$ and for all $c \in t_C$, $\bar{\beta}_c(\sigma_{\mathcal{I}}) = \beta_c$. Therefore, for all $i \in t$, $\bar{\beta}_i(\sigma_{\mathcal{I}}) > \bar{\alpha}_i(\sigma_{\mathcal{I}})$ and $\bar{u}_i(\sigma_{\mathcal{I}}) > 0$. Furthermore, it follows from Lemma 11 that for all $\sigma'_t \in \Sigma_t \setminus \Sigma_t(\sigma_{\mathcal{I}})$, $\bar{\beta}_i(\sigma'_t, \sigma_{\mathcal{I} \setminus t}) = 0$. Therefore, $\bar{u}_i(\sigma'_t, \sigma_{\mathcal{I} \setminus t}) \leq 0 < \bar{u}_i(\sigma_{\mathcal{I}})$, which implies that $\sigma_{\mathcal{I}} \succ_t (\sigma'_t, \sigma_{\mathcal{I} \setminus t})$. Consequently, for all $\sigma_t^* \in \Sigma_t(\sigma_{\mathcal{I}})$, if $(\sigma_t^*, \sigma_{\mathcal{I} \setminus t}) \succeq_t \sigma_{\mathcal{I}}$, then $(\sigma_t^*, \sigma_{\mathcal{I} \setminus t}) \succ_t (\sigma'_t, \sigma_{\mathcal{I} \setminus t})$. This allows us to conclude that $\sigma_{\mathcal{I}}$ is $(N_{\mathcal{T}}^P + N_{\mathcal{T}}^C, N_{\mathcal{T}}^P, N_{\mathcal{T}}^C)$ -cotolerant. \square

4.4 Discussion

Some important consequences result from Theorems 9 and 12. One is that $\sigma_{\mathcal{I}}$ is $(1, 1, 1)$ -cotolerant. By the definition of \succeq_t for any $t \in \mathcal{T}$ such that $\#t = 1$ and by the fact that $\Sigma_t(\sigma_{\mathcal{I}}) = \{\sigma_{\mathcal{I}}\}$, $\sigma_{\mathcal{I}}$ is a Nash equilibrium.

Another important result is that no producer $p \in t_P$ from any non-Byzantine coalition t can avoid sending the expected SUMMARY and BLOCK messages to consumers not from t_C . The same applies to REPORT messages sent by consumers to TO. Therefore, for any $i \in t$, the expected utility of delaying messages to players not from t is at most as high as the utility of following the algorithm. Therefore, by the promptness principle, players never delay messages between different coalitions. Concerning the messages exchanged between players from the same coalition, we do not guarantee that players do not incur any communication delays. Though, if these messages are mandatory to ensure that all players of the coalition are rewarded, then, if there is any delay, it must be finite, otherwise, the expected utility is the same as not sending these messages, i.e., at most 0.

Acknowledgements

This work was partially supported by the FCT (INESC-ID multi annual funding through the PIDDAC Program fund grant and by the project PTDC/EIA-EIA/102212/2008).

References

1. Anderson, D.: Boinc: A system for public-resource computing and storage. In: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing. GRID'04, Pittsburgh, PA, USA, IEEE (November 2004) 4–10
2. Aiyer, S., Alvisi, L., Clement, A., Dahlin, M., Martin, J.P., Porth, C.: BAR fault tolerance for cooperative services. In: Proceedings of the 20th ACM Symposium on Operating Systems Principles. SOSP'05, Brighton, United Kingdom, ACM (October 2005) 45–58
3. Vilaça, X., Leitão, J., Correia, M., Rodrigues, L.: N-party BAR transfer. In: Proceedings of the 15th International Conference On Principles Of Distributed Systems (to appear). OPODIS'11, Toulouse, France (December 2011)
4. Eliaz, K.: Fault-tolerant implementation. *Review of Economic Studies* **69**(3) (August 2002) 589–610
5. Moscibroda, T., Schmid, S., Wattenhofer, R.: On the topologies formed by selfish peers. In: Proceedings of the 25th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing. PODC'06, Denver, CO, USA, ACM (July 2006) 133–142
6. Wong, E.L., Clement, A., Levy, I., Alvisi, L., Dahlin, M.: Regret freedom isn't free. In: Proceedings of the 15th International Conference On Principles Of Distributed Systems (to appear). OPODIS'11, Toulouse, France (December 2011)
7. Aumann, R.J.: Acceptable points in General Cooperative n -person Games. In: Contributions to the Theory of Games IV. Number 40 in Annals of Mathematics Studies. Princeton University Press, Princeton (1959) 287–324
8. Bernheim, B., Peleg, B., Whinston, M.: Coalition-proof nash equilibria i. concepts. *Journal of Economic Theory* **42**(1) (June 1987) 1–12
9. Moreno, D., Wooders, J.: Coalition-proof equilibrium. *Games and Economic Behavior* **17**(1) (November 1996) 80–112
10. Abraham, I., Dolev, D., Gonen, R., Halpern, J.: Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In: Proceedings of the 25th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing. PODC'06, Denver, CO, USA, ACM (July 2006) 53–62
11. Clement, A., Napper, J., Li, H., Martin, J.P., Alvisi, L., Dahlin, M.: Theory of BAR games. In: Proceedings of the 26th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing. PODC'07, Portland, OR, USA, ACM (August 2007) 358–359
12. Li, H., Clement, A., Wong, E., Napper, J., Roy, I., Alvisi, L., Dahlin, M.: BAR gossip. In: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation. OSDI'06, Seattle, WA, USA, USENIX Association (November 2006) 191–204
13. Li, H., Clement, A., Marchetti, M., Kapritsos, M., Robison, L., Alvisi, L., Dahlin, M.: Flightpath: Obedience vs choice in cooperative services. In: Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation. OSDI'08, San Diego, CA, USA, USENIX Association (December 2008) 355–368
14. Mokhtar, S., Pace, A., Quéma, V.: FireSpam: Spam resilient gossiping in the BAR model. In: Proceedings of the 29th IEEE International Symposium on Reliable Distributed Systems. SRDS'10, New Delhi, India, IEEE (October 2010) 225–234
15. Wong, E.L., Leners, J.B., Alvisi, L.: It's on me! the benefit of altruism in BAR environment. In: Proceedings of the 25th International Symposium on Distributed Computing. DISC'10, Cambridge, USA, Springer (September 2010) 406–420
16. Cachin, C., Guerraoui, R., Rodrigues, L.: Introduction to Reliable and Secure Distributed Programming. 2nd edition edn. Springer-Verlag New York, Inc. (2011)